DOCUMENT-IDENTIFIER: US 20020156981 A1
TITLE: Cacheable above one megabyte system management random access memory

RSM

Summary of Invention Paragraph:
[0008] The only way to exit from the system management mode is to execute the resume instruction. The resume instruction is only available in SMM mode. The resume instruction restores the processor's context by loading the state save image from SMRAM back into the processor's registers. It then returns program control back to the interrupted or foreground program code.

Summary of Invention Paragraph:
[0017] The present invention un-caches the SMRAM while operating outside of system management mode (SMM). When an SMI is generated, the CPU operation is transferred to the system management mode. The SMI dispatcher changes cache settings to cache the extended memory and the SMRAM with write-through. The SMI dispatcher caches the extended memory with write-back and un-caches the SMRAM upon generation of a resume instruction (exit system management interrupt, or exit SMI) which exits the system management mode.

Detail Description Paragraph:
[0032] Referring to FIG. 3, it is a flow diagram illustrating steps in implementing an exemplary method 30 in accordance with the principles of the present invention that provides for cacheable above one megabyte system management random access memory 27. While operating outside of system management mode (SMM), the SMRAM 27 is un-cached 31. When an SMI is executed 32, the CPU 11 was transferred to the system management mode and it executes SMI dispatcher 12. The SMI dispatcher 12 changes 34 cache settings to cache the extended memory 28 and SMRAM 27 with write-through. After the SMI event is serviced, the SMI dispatcher 12 changes 36 cache settings to cache the extended memory 28 with write-back and un-cache the SMRAM 27, then resume instruction is executed 35 which exits the system management mode.

CLAIMS:

1. A method for use in a computer system having a central processing unit (CPU) having a processor and a system management interrupt dispatcher, a cache coupled to the CPU, and a chipset memory controller that interfaces the CPU to a memory comprising system memory and system management random access memory (SMRAM), the method comprising the steps of: un-caching the SMRAM while operating outside of system management mode; transferring CPU operation to system management mode in response to execution of a system management interrupt; changing cache settings to cache the extended memory and system management random access memory with write-through; changing cache settings to cache the extended memory with write-back and uncache the SMRAM; and executing a resume instruction to exit SMM mode.

6. A computer system comprising: a central processing unit (CPU) comprising a processor and a system management interrupt dispatcher; a cache coupled to the CPU; and a chipset memory controller that interfaces the CPU to a memory comprising extended memory and system management random access memory (SMRAM); wherein the system management interrupt dispatcher un-caches the SMRAM while operating outside of system management mode, transfers CPU operation to system management mode in response to a system management interrupt, changes cache settings to cache the system memory and system management random access memory with write-through, changes cache settings to cache the system memory with write-back and un-cache the SMRAM, and executes resume instruction to exit SMM mode.

Generate Collection | Print

L44: Entry 11 of 111 File: PGPB Feb 6, 2003

PGPUB-DOCUMENT-NUMBER: 20030028781
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030028781 A1

TITLE: Mechanism for closing back door access mechanisms in personal computer systems

PUBLICATION-DATE: February 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Strongin, Geoffrey S.	Austin	TX	US	

APPL-NO: 09/ 853226 [PALM]
DATE FILED: May 11, 2001

RELATED-US-APPL-DATA:
Application 09/853226 is a continuation-in-part-of US application 09/852372, filed May 10, 2001, PENDING
Application 09/853226 is a continuation-in-part-of US application 09/852942, filed May 10, 2001, PENDING

INT-CL: [07] H04 K 1/00

US-CL-PUBLISHED: 713/182
US-CL-CURRENT: 713/182

REPRESENTATIVE-FIGURES: 3

ABSTRACT:

Methods, devices, and systems for closing back door access mechanisms. A processor includes a first register configured to store one or more hardware-debug-test (HDT) enable bits, a first control logic coupled to receive a plurality of HDT input signals, and a second control logic coupled to the first register. The first control logic is coupled to access the first register. The second control logic is configured to store one or more default values in the first register in response to a reset of the processor. Another processor includes a first control logic coupled to receive a plurality of microcode inputs, a first register coupled to the first control logic, and a second control logic coupled to the first register. The first register is configured to store one or more microcode loader enable bits. The second control logic is configured to store one or more default values in the first register in response to a reset of the processor.

[0001] This Application is a continuation-in-part of co-pending U.S. patent application Ser. No. _____, entitled, "Secure Execution Box and Method," filed on May 10, 2001, whose inventors are Dale E. Gulick and Geoffrey S. Strongin. This Application is also a continuation-in-part of co-pending U.S. patent application Ser. No. _____, entitled, "Computer System Architecture for Enhanced Security and Manageability," filed on May 10, 2001, whose inventors are Geoffrey S. Strongin and Dale E. Gulick.

L44: Entry 11 of 111 File: PGPB Feb 6, 2003

DOCUMENT-IDENTIFIER: US 20030028781 A1
TITLE: Mechanism for closing back door access mechanisms in personal computer systems

Detail Description Paragraph:
[0086] The SMM timing controller 401A includes the duration timer 406A, which measures how long the computer system 100 is in SMM. The kick-out timer 407A, also included in the SMM timing controller 401A, counts down from a predetermined value while the computer system 100 is in SMM. The control logic 420A is configured to assert a control signal (EXIT SMM 404) for the processor to exit SMM, such as in response to the expiration of the kick-out timer 407A. The restart timer 408, included in the SMM timing controller 401A, starts counting down from a predetermined value after the kick-out timer 407A reaches zero. The SMM indicator 405, also included in the SMM timing controller 401 A, is operable to monitor the status of one or more signals in the computer system, such as the SMI# (System Management Interrupt) signal and/or the SMIACT# (SMI ACTive) signal to determine if the computer system is in SMM.

Detail Description Paragraph:
[0091] The SMM timing controller 401B includes the duration/kick-out timer 407B measures how long the computer system 100 is in SMM, counting up to a predetermined value while the computer system 100 is in SMM. The control logic 420B is configured to assert a control signal for the processor to exit SMM in response to the duration/kick-out timer 407B reaching the predetermined value. The restart timer 408 starts counting down from a predetermined value after the duration/kick-out timer 407B reaches the predetermined value. The SMM indicator 405 is operable to monitor the status of one or more signals in the computer system, such as the SMI# (System Management Interrupt) signal and/or the SMIACT# (SMI ACTive) signal, to determine if the computer system is in SMM.

Detail Description Paragraph:
[0133] FIGS. 9A and 9B illustrate block diagrams of embodiments of computer systems 800A and 800B that control the timing and duration of SMM, according to various aspects of the present invention. FIGS. 9A and 9B include a processor 805, a north bridge 810, memory 106, and the south bridge 330. The processor includes an SMM exit controller 807 and one or more SMM MSRs (machine specific registers) 807. The north bridge 810 includes a memory controller 815. The south bridge 330 includes the SMM timing controller 401 and the scratchpad RAM 440. The north bridge 810 is coupled between the processor 805 and the south bridge 330, to the processor 805 through a local bus 808 and to the south bridge 330 through the PCI bus 110. The north bridge 810 is coupled to receive the SMIACT# signal from the processor 805.

Detail Description Paragraph:
[0134] In the embodiment of FIG. 9A, the computer system 800A signals that the processor 805 is in SMM using standard processor signals (e.g. SMIACT# to the north bridge 810) and/or bus cycles on the local bus 808 and PCI bus 110. In the embodiment of FIG. 9B, the computer system 800B signals that the processor 805 is in SMM using standard processor signals (e.g. SMIACT#) to both the north bridge 810 and the south bridge 330. An exit SMM signal 404 is also shown between the SMM timing controller 401 and the SMM exit controller 806.

Detail Description Paragraph:
[0138] The SMM exit controller 806 in the processor 805 is configured to receive a request to the processor 805 to exit SMM. In one embodiment, the SMM exit controller 806 is operable to exit SMM prior to completing the task for which the SMI# was originally asserted that led to the processor 805 being in SMM. Upon receiving the request to exit SMM, the SMM exit controller 806 is configured to read the contents of the one or more SMM MSRs 807 to obtain a jump location for a clean-up routine, preferably stored in ROM, in SMM memory space. The SMM MSRs 807 may also store one or more bits to indicate that an SMM routine has been interrupted and/or a re-entry point (e.g. an address in SMM memory space) in the interrupted SMM routine. The SMM

exit controller 806 may be configured to store the one or more bits indicating that the SMM routine has been interrupted and the re-entry point.

Detail Description Paragraph:
[0140] The method next checks to determine if the kick-out timer 407 has expired in decision block 915. If the kick-out timer 407 has not expired, then the method continues checking to determine if the kick-out timer 407 has expired in decision block 915. If the kick-out timer 407 has expired in decision block 915, then the method transmits a request to the processor to exit SMM without completing the SMI request that invoked SMM, in block 920. The processor saves the state of the SMM session without finishing the SMM session and exits SMM, in block 925.

Detail Description Paragraph:
[0141] The request to the processor to exit SMM, in block 920, may include submitting an RSM (Resume from System Management mode) instruction, or other control signal delivered over the system bus, to the processor. Upon executing the RSM instruction, or receiving the control signal through the interface logic to the system bus, the processor exits SMM and the processor's previous state is restored from system management memory. The processor then resumes any application that was interrupted by SMM. In another embodiment, the request to the processor to exit SMM includes another device in the computer system, such as the south bridge, asserting a control signal, such as the exit SMM signal, to the processor to exit SMM.

Detail Description Paragraph:
[0145] FIGS. 11A and 11B illustrate flowcharts of embodiments of methods 1100A and 1100B for upgrading the monotonic counter 435B, which may be stored in the SMM ROM 550, according to various aspects of the present invention. The method 1100A, shown in FIG. 11A, includes checking the RTC checksum, in block 1105. In decision block 1110, if the RTC checksum is valid, then the method 1100A exits. In decision block 1110, if the RTC checksum is not valid, then the method 1100 inspects the monotonic counter 435B in the SMM ROM 550 in block 1115. In decision block 1 120A, the method checks if the value stored in the monotonic counter 435B in the SMM ROM 550 is the default (e.g. reset or rollover) value.

Detail Description Paragraph:
[0147] The method 1110B, shown in FIG. 11B, includes checking the RTC checksum, in block 1105. In decision block 1110, if the RTC checksum is valid, then the method 1100A exits. In decision block 1110, if the RTC checksum is not valid, then the method 1100 inspects the monotonic counter 435B in the SMM ROM 550 in block 1115. In decision block 1120B, the method checks if the values stored in the monotonic counter 435B in the SMM ROM 550 are all ones.

Detail Description Paragraph:
[0149] FIGS. 12A and 12B illustrate flowcharts of embodiments methods 1200A and 1200B for updating a monotonic counter 435A in the south bridge 330, according to various aspects of the present invention. The method 1200A checks to see if the value stored in the monotonic counter 435A in the south bridge 330 is the maximum value that can be stored, in decision block 1205A. If the value stored in the monotonic counter 435A in the south bridge 330 is not the maximum value, in decision block 1205, then the method 1200A exits. If the value stored in the monotonic counter 435A in the south bridge 330 is the maximum value that can be stored, in decision block 1205, then the method 1200A inspects the monotonic counter 435B in the SMM ROM 550 in decision block 1210. The method 1200A checks to see if the value stored in the monotonic counter 435B in the SMM ROM 550 is the default (or reset) value, in decision block 1215A.

Detail Description Paragraph:
[0151] The method 1200B, shown in FIG. 12B, checks to see if all values in the monotonic counter 435A in the south bridge 330 are equal to one (i.e. the reset value), in decision block 1205B. If all values in the monotonic counter 435A in the south bridge 330 are not equal to one, in decision block 1205B, then the method 1200B exits. If all values in the monotonic counter 435A in the south bridge 330 are equal to one, in decision block 1205B, then the method 1200B inspects the monotonic counter 435B in the SMM ROM 550, in decision block 1210. The method 1200B checks to see if all values in the monotonic counter 435B in the SMM ROM 550 are equal to one, in decision block 1215B.

Detail Description Paragraph:
[0179] Turning now to FIG. 16D, the method 1600D includes a processor, such as processors 102, 805, etc., operating in a mode that is not SMM, in block 1604. In block 1606, code being processed by the processor attempts to access any part of the security hardware 370, or other

hardware whose access may require a check of an access lock similar to the access locks 460. The method checks, at decision block 1607, to see if the security hardware 370 is available. If the security hardware 370 is not available, at decision block 1607, then the method 1600D exits or returns. If the security hardware 370 is available, at decision block 1607, then the method 1660D accesses the security hardware 370, at block 1630. The method, optionally, closes the access locks to the security hardware, if necessary, at block 1650.

Detail Description Paragraph:
[0193] FIG. 18A illustrates a prior art flowchart of an SMM program 1800A. The prior art SMM program 1800A starts at 1805, includes one or more instructions for execution in SMM, in block 1810A, and ends at 1895 without interruption. In other words, prior art SMM program 1800A is uninterruptible and has no other entry points than the start at 1805. There are also no reasonable exit points, barring processor failure, other than the stop at 1895.

Detail Description Paragraph:
[0194] FIG. 18B illustrate a flowchart of an embodiment of operations of an interruptible and re-enterable SMM program 1800B, according to one aspect of the present invention. In contrast to the prior art SMM program 1800A, the interruptible and re-enterable SMM program 1800B includes a start at 1805, one or more instructions for execution in SMM, in block 1810B, an entry/exit point 1815, one or more instructions for execution in SMM, in block 1820, and the stop at 1895.

Detail Description Paragraph:
[0195] Also in contrast to the prior art SMM program 1800A, FIG. 18C illustrates an embodiment of operation of a computer system running the interruptible and re-enterable SMM program 1800B, according to one aspect of the present invention. The operations 1800C of the computer system includes a start 1805. The operations also include receiving a request to enter SMM, at 1810 and saving the system state at 1815. The method checks, at 1820, for a saved SMM state, as could be found from exiting the SMM program 1800B at 1875. If no saved SMM state is found at 1820, then load the requested default SMM state at 1825. If a saved SMM state is found at 1820, then load the saved SMM state, at 1830.

Detail Description Paragraph:
[0196] The method 1800C executes the loaded SMM state, at 1835, either the default state from 1825 or the saved state at 1830. If the SMM processing is finished, at 1840, then the method moves to 1855 and exits SMM. If the SMM processing is not finished, then the method continues execution of the SMM state, if no exit request is received at 1845. If the exit request is received at 1845, then the method saves the current SMM state at 1850 and exits SMM at 1855. The saved system state is reloaded at 1860, and the method ends at the stop 1895.

Detail Description Paragraph:
[0197] While only one entry/exit point 1815 is shown in the embodiment of FIG. 18B, other embodiments may include two or more entry/exit points 1815 in an SMM program 1800B or the operations of the method 1800C shown in FIG. 18C. In these embodiments, each entry/exit point 1815 would have one or more instructions for execution in SMM, similar to blocks 1810B and 1820, both before and after the entry/exit point 1815.

Detail Description Paragraph:
[0198] For example, in one embodiment, block 1810B includes one instruction for execution in SMM, followed by an entry/exit point 1815A. Entry/exit point 1815A is followed by another single instruction for execution in SMM, in block 1820A. Block 1820A is followed by another entry/exit point 1815B. Entry/exit point 1815B is followed by another single instruction for execution in SMM, in block 1820B. Block 1820B is followed by the stop 1895. While a single instruction in blocks 1810B, 1820A, and 1820B may be small, the concept of regularly spaced Entry/exit points 1815 is illustrated. In other embodiments, two, three or more instructions for execution in SMM may be substituted for the single instructions. In still other embodiments, there may be a variable number of instructions for execution in SMM in blocks 1810B, and 1820. The number of instructions may depend on the execution times for each set of instructions, so that SMM may be interruptible every so often during execution.

Detail Description Paragraph:
[0199] It is noted that forced exits from SMM, as are taught herein in one aspect of the present invention, for example, with respect to FIG. 10A, and re-entry into SMM, as is also taught herein in another aspect of the present invention, for example, with respect to FIG.

10B, are but two examples of how interruptible, re-enterable SMM code could be implemented or used. Those of skill in the art of computer programming with full appreciation of this disclosure will appreciate that many programming techniques used with non-SMM code that used interruptible, re-enterable code flow will now be available in SMM code.

Detail Description Paragraph:
[0301] In FIG. 38A, the processor executes BIOS code instructions from SMM space, in 4920. After optionally accessing the security hardware, in 4930, the method 4900A requests authentication from the crypto-processor, preferably using the master mode logic, in 4835A. The method 4900A places the bus interface logics in master mode, in 4938. The bus interface logics would typically be between the crypto-processor and the authentication device. The method 4900A receives the authentication data while the bus interface logics are in master mode, in 4940. The method 4900A exits master mode and flushes the buffers of the bus interface logics, in 4942. The method 4900A next verifies the authentication data, in 4944. Verifying the authentication data may include the crypto-processor providing an indication of the authentication data to a remote security device. If the authentication data are verified in 4948, then the method 4900A continues the boot process, in 4990. If the authentication data are not verified in 4948, then the method 4900A returns to 4935A and again requests authentication.

Detail Description Paragraph:
[0302] In FIG. 38B, the processor executes BIOS code instructions from SMM space, in 4920. After optionally accessing the security hardware, in 4930, and optionally entering a BIOS management mode, in 4932, the method 4900B requests authentication from the security hardware, using the master mode logic, in 4935B. The method 4900B places the bus interface logics in master mode, in 4938. The bus interface logics would typically be between the security hardware, e.g. the south bridge, and the authentication device. The method 4900B receives the authentication data while the bus interface logics are in master mode, in 4940. The method 4900B exits master mode and flushes the buffers of the bus interface logics, in 4942. The method 4900B next verifies the authentication data, in 4944. Verifying the authentication data may include the security hardware providing an indication of the authentication data to a remote security device. If the authentication data are verified in 4948, then the method 4900B continues the boot process, in 4990. If the authentication data are not verified in 4948, then the method 4900B returns to 4935A and again requests authentication.

First Hit

Generate Collection Print

L44: Entry 3 of 111 File: PGPB Dec 11, 2003

PGPUB-DOCUMENT-NUMBER: 20030229794
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030229794 A1

TITLE: System and method for protection against untrusted system management code by redirecting a system management interrupt and creating a virtual machine container

PUBLICATION-DATE: December 11, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Sutton, James A. II	Portland	OR	US	
Grawrock, David W.	Aloha	OR	US	
Uhlig, Richard A.	Hillsboro	OR	US	
Poisner, David I.	Folsom	CA	US	
Glew, Andrew F.	Portland	OR	US	
Hall, Clifford D.	Orangevale	CA	US	
Smith, Lawrence O. III	Beaverton	OR	US	
Neiger, Gilbert	Portland	OR	US	
Kozuch, Michael A.	Export	PA	US	
George, Robert T.	Austin	TX	US	
Burgess, Bradley G.	Austin	TX	US	

APPL-NO: 10/ 165597 [PALM]
DATE FILED: June 7, 2002

INT-CL: [07] G06 F 12/14

US-CL-PUBLISHED: 713/189; 713/164
US-CL-CURRENT: 713/189; 713/164

REPRESENTATIVE-FIGURES: 2

ABSTRACT:

A system and method for permitting the execution of system management mode (SMM) code during secure operations in a microprocessor system is described. In one embodiment, the system management interrupt (SMI) may be first directed to a handler in a secured virtual machine monitor (SVMM). The SMI may then be re-directed to SMM code located in a virtual machine (VM) that is under the security control of the SVMM. This redirection may be accomplished by allowing the SVMM to read and write the system management (SM) base register in the processor.

L44: Entry 10 of 111 File: PGPB Feb 27, 2003

DOCUMENT-IDENTIFIER: US 20030041248 A1
TITLE: External locking mechanism for personal computer memory locations

Detail Description Paragraph:
[0082] The SMM timing controller 401A includes the duration timer 406A, which measures how long the computer system 100 is in SMM. The kick-out timer 407A, also included in the SMM timing controller 401A, counts down from a predetermined value while the computer system 100 is in SMM. The control logic 420A is configured to assert a control signal (EXIT SMM 404) for the processor to exit SMM, such as in response to the expiration of the kick-out timer 407A. The restart timer 408, included in the SMM timing controller 401A, starts counting down from a predetermined value after the kick-out timer 407A reaches zero. The SMM indicator 405, also included in the SMM timing controller 401A, is operable to monitor the status of one or more signals in the computer system, such as the SMI# (System Management Interrupt) signal and/or the SMIACT# (SMI ACTive) signal to determine if the computer system is in SMM.

Detail Description Paragraph:
[0087] The SMM timing controller 401B includes the duration/kick-out timer 407B measures how long the computer system 100 is in SMM; counting up to a predetermined value while the computer system 100 is in SMM. The control logic 420B is configured to assert a control signal for the processor to exit SMM in response to the duration/kick-out timer 407B reaching the predetermined value. The restart timer 408 starts counting down from a predetermined value after the duration/kick-out timer 407B reaches the predetermined value. The SMM indicator 405 is operable to monitor the status of one or more signals in the computer system, such as the SMI# (System Management Interrupt) signal and/or the SMIACT# (SMI ACTive) signal, to determine if the computer system is in SMM.

Detail Description Paragraph:
[0129] FIGS. 9A and 9B illustrate block diagrams of embodiments of computer systems 800A and 800B that control the timing and duration of SMM, according to various aspects of the present invention. FIGS. 9A and 9B include a processor 805, a north bridge 810, memory 106, and the south bridge 330. The processor includes an SMM exit controller 807 and one or more SMM MSRs (machine specific registers) 807. The north bridge 810 includes a memory controller 815. The south bridge 330 includes the SMM timing controller 401 and the scratchpad RAM 440. The north bridge 810 is coupled between the processor 805 and the south bridge 330, to the processor 805 through a local bus 808 and to the south bridge 330 through the PCI bus 110. The north bridge 810 is coupled to receive the SMIACT# signal from the processor 805.

Detail Description Paragraph:
[0130] In the embodiment of FIG. 9A, the computer system 800A signals that the processor 805 is in SMM using standard processor signals (e.g. SMIACT# to the north bridge 810) and/or bus cycles on the local bus 808 and PCI bus 110. In the embodiment of FIG. 9B, the computer system 800B signals that the processor 805 is in SMM using standard processor signals (e.g. SMIACT#) to both the north bridge 810 and the south bridge 330. An exit SMM signal 404 is also shown between the SMM timing controller 401 and the SMM exit controller 806.

Detail Description Paragraph:
[0134] The SMM exit controller 806 in the processor 805 is configured to receive a request to the processor 805 to exit SMM. In one embodiment, the SMM exit controller 806 is operable to exit SMM prior to completing the task for which the SMI# was originally asserted that led to the processor 805 being in SMM. Upon receiving the request to exit SMM, the SMM exit controller 806 is configured to read the contents of the one or more SMM MSRs 807 to obtain a jump location for a clean-up routine, preferably stored in ROM, in SMM memory space. The SMM MSRs 807 may also store one or more bits to indicate that an SMM routine has been interrupted and/or a re-entry point (e.g. an address in SMM memory space) in the interrupted SMM routine. The SMM

exit controller 806 may be configured to store the one or more bits indicating that the SMM routine has been interrupted and the re-entry point.

Detail Description Paragraph:
[0136] The method next checks to determine if the kick-out timer 407 has expired in decision block 915. If the kick-out timer 407 has not expired, then the method continues checking to determine if the kick-out timer 407 has expired in decision block 915. If the kick-out timer 407 has expired in decision block 915, then the method transmits a request to the processor to exit SMM without completing the SMI request that invoked SMM, in block 920. The processor saves the state of the SMM session without finishing the SMM session and exits SMM, in block 925.

Detail Description Paragraph:
[0137] The request to the processor to exit SMM, in block 920, may include submitting an RSM (Resume from System Management mode) instruction, or other control signal delivered over the system bus, to the processor. Upon executing the RSM instruction, or receiving the control signal through the interface logic to the system bus, the processor exits SMM and the processor's previous state is restored from system management memory. The processor then resumes any application that was interrupted by SMM. In another embodiment, the request to the processor to exit SMM includes another device in the computer system, such as the south bridge, asserting a control signal, such as the exit SMM signal, to the processor to exit SMM.

Detail Description Paragraph:
[0141] FIGS. 11A and 11B illustrate flowcharts of embodiments of methods 1100A and 1100B for upgrading the monotonic counter 435B, which may be stored in the SMM ROM 550, according to various aspects of the present invention. The method 110A, shown in FIG. 11A, includes checking the RTC checksum, in block 1105. In decision block 1110, if the RTC checksum is valid, then the method 1100A exits. In decision block 1110, if the RTC checksum is not valid, then the method 1100 inspects the monotonic counter 435B in the SMM ROM 550 in block 1115. In decision block 1120A, the method checks if the value stored in the monotonic counter 435B in the SMM ROM 550 is the default (e.g. reset or rollover) value.

Detail Description Paragraph:
[0143] The method 1100B, shown in FIG. 11B, includes checking the RTC checksum, in block 1105. In decision block 1110, if the RTC checksum is valid, then the method 1100A exits. In decision block 1110, if the RTC checksum is not valid, then the method 1100 inspects the monotonic counter 435B in the SMM ROM 550 in block 1115. In decision block 1120B, the method checks if the values stored in the monotonic counter 435B in the SMM ROM 550 are all ones.

Detail Description Paragraph:
[0145] FIGS. 12A and 12B illustrate flowcharts of embodiments methods 1200A and 1200B for updating a monotonic counter 435A in the south bridge 330, according to various aspects of the present invention. The method 1200A checks to see if the value stored in the monotonic counter 435A in the south bridge 330 is the maximum value that can be stored, in decision block 1205A. If the value stored in the monotonic counter 435A in the south bridge 330 is not the maximum value, in decision block 1205, then the method 1200A exits. If the value stored in the monotonic counter 435A in the south bridge 330 is the maximum value that can be stored, in decision block 1205, then the method 1200A inspects the monotonic counter 435B in the SMM ROM 550 in decision block 1210. The method 1200A checks to see if the value stored in the monotonic counter 435B in the SMM ROM 550 is the default (or reset) value, in decision block 1215A.

Detail Description Paragraph:
[0147] The method 1200B, shown in FIG. 12B, checks to see if all values in the monotonic counter 435A in the south bridge 330 are equal to one (i.e. the reset value), in decision block 1205B. If all values in the monotonic counter 435A in the south bridge 330 are not equal to one, in decision block 1205B, then the method 1200B exits. If all values in the monotonic counter 435A in the south bridge 330 are equal to one, in decision block 1205B, then the method 1200B inspects the monotonic counter 435B in the SMM ROM 550, in decision block 1210. The method 1200B checks to see if all values in the monotonic counter 435B in the SMM ROM 550 are equal to one, in decision block 1215B.

Detail Description Paragraph:
[0175] Turning now to FIG. 16D, the method 1600D includes a processor, such as processors 102, 805, etc., operating in a mode that is not SMM, in block 1604. In block 1606, code being processed by the processor attempts to access any part of the security hardware 370, or other

hardware whose access may require a check of an access lock similar to the access locks 460. The method checks, at decision block 1607, to see if the security hardware 370 is available. If the security hardware 370 is not available, at decision block 1607, then the method 1600D exits or returns. If the security hardware 370 is available, at decision block 1607, then the method 1660D accesses the security hardware 370, at block 1630. The method, optionally, closes the access locks to the security hardware, if necessary, at block 1650.

Detail Description Paragraph:
[0189] FIG. 18A illustrates a prior art flowchart of an SMM program 1800A. The prior art SMM program 1800A starts at 1805, includes one or more instructions for execution in SMM, in block 1810A, and ends at 1895 without interruption. In other words, prior art SMM program 1800A is uninterruptible and has no other entry points than the start at 1805. There are also no reasonable exit points, barring processor failure, other than the stop at 1895.

Detail Description Paragraph:
[0190] FIG. 18B illustrate a flowchart of an embodiment of operations of an interruptible and re-enterable SMM program 1800B, according to one aspect of the present invention. In contrast to the prior art SMM program 1800A, the interruptible and re-enterable SMM program 1800B includes a start at 1805, one or more instructions for execution in SMM, in block 1810B, an entry/exit point 1815, one or more instructions for execution in SMM, in block 1820, and the stop at 1895.

Detail Description Paragraph:
[0191] Also in contrast to the prior art SMM program 1800A, FIG. 18C illustrates an embodiment of operation of a computer system running the interruptible and re-enterable SMM program 1800B, according to one aspect of the present invention. The operations 1800C of the computer system includes a start 1805. The operations also include receiving a request to enter SMM, at 1810 and saving the system state at 1815. The method checks, at 1820, for a saved SMM state, as could be found from exiting the SMM program 1800B at 1875. If no saved SMM state is found at 1820, then load the requested default SMM state at 1825. If a saved SMM state is found at 1820, then load the saved SMM state, at 1830.

Detail Description Paragraph:
[0192] The method 1800C executes the loaded SMM state, at 1835, either the default state from 1825 or the saved state at 1830. If the SMM processing is finished, at 1840, then the method moves to 1855 and exits SMM. If the SMM processing is not finished, then the method continues execution of the SMM state, if no exit request is received at 1845. If the exit request is received at 1845, then the method saves the current SMM state at 1850 and exits SMM at 1855. The saved system state is reloaded at 1860, and the method ends at the stop 1895.

Detail Description Paragraph:
[0193] While only one entry/exit point 1815 is shown in the embodiment of FIG. 18B, other embodiments may include two or more entry/exit points 1815 in an SMM program 1800B or the operations of the method 1800C shown in FIG. 18C. In these embodiments, each entry/exit point 1815 would have one or more instructions for execution in SMM, similar to blocks 1810B and 1820, both before and after the entry/exit point 1815.

Detail Description Paragraph:
[0194] For example, in one embodiment, block 1810B includes one instruction for execution in SMM, followed by an entry/exit point 1815A. Entry/exit point 1815A is followed by another single instruction for execution in SMM, in block 1820A. Block 1820A is followed by another entry/exit point 1815B. Entry/exit point 1815B is followed by another single instruction for execution in SMM, in block 1820B. Block 1820B is followed by the stop 1895. While a single instruction in blocks 1810B, 1820A, and 1820B may be small, the concept of regularly spaced Entry/exit points 1815 is illustrated. In other embodiments, two, three or more instructions for execution in SMM may be substituted for the single instructions. In still other embodiments, there may be a variable number of instructions for execution in SMM in blocks 1810B, and 1820. The number of instructions may depend on the execution times for each set of instructions, so that SMM may be interruptible every so often during execution.

Detail Description Paragraph:
[0195] It is noted that forced exits from SMM, as are taught herein in one aspect of the present invention, for example, with respect to FIG. 10A, and re-entry into SNM, as is also taught herein in another aspect of the present invention, for example, with respect to FIG.

10B, are but two examples of how interruptible, re-enterable SMM code could be implemented or used. Those of skill in the art of computer programming with full appreciation of this disclosure will appreciate that many programming techniques used with non-SMM code that used interruptible, re-enterable code flow will now be available in SMM code.

Detail Description Paragraph:
[0297] In FIG. 38A, the processor executes BIOS code instructions from SMM space, in 4920. After optionally accessing the security hardware, in 4930, the method 4900A requests authentication from the crypto-processor, preferably using the master mode logic, in 4835A. The method 4900A places the bus interface logics in master mode, in 4938. The bus interface logics would typically be between the crypto-processor and the authentication device. The method 4900A receives the authentication data while the bus interface logics are in master mode, in 4940. The method 4900A exits master mode and flushes the buffers of the bus interface logics, in 4942. The method 4900A next verifies the authentication data, in 4944. Verifying the authentication data may include the crypto-processor providing an indication of the authentication data to a remote security device. If the authentication data are verified in 4948, then the method 4900A continues the boot process, in 4990 If the authentication data are not verified in 4948, then the method 4900A returns to 4935A and again requests authentication.

Detail Description Paragraph:
[0298] In FIG. 38B, the processor executes BIOS code instructions from SMM space, in 4920. After optionally accessing the security hardware, in 4930, and optionally entering a BIOS management mode, in 4932, the method 4900B requests authentication from the security hardware, using the master mode logic, in 4935B. The method 4900B places the bus interface logics in master mode, in 4938. The bus interface logics would typically be between the security hardware, e.g the south bridge, and the authentication device. The method 4900B receives the authentication data while the bus interface logics are in master mode, in 4940. The method 4900B exits master mode and flushes the buffers of the bus interface logics, in 4942. The method 4900B next verifies the authentication data, in 4944. Verifying the authentication data may include the security hardware providing an indication of the authentication data to a remote security device. If the authentication data are verified in 4948, then the method 4900B continues the boot process, in 4990. If the authentication data are not verified in 4948, then the method 4900B returns to 4935A and again requests authentication.

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, April 16, 2004

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
☐	L45	(713/182).ccls.	362
☐	L44	exit$3 same L42	111
☐	L43	exit$3 and L42	440
☐	L42	L1 or smm	1813
	DB=PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR		
☐	L41	L17 and L15	0
☐	L40	L25 and L10	0
☐	L39	L22 and L10	0
	DB=USPT; PLUR=YES; OP=OR		
☐	L38	5012514.pn.	1
☐	L37	5377269.pn.	1
☐	L36	5537544.pn.	1
☐	L35	5586301.pn.	1
☐	L34	5596718.pn.	1
☐	L33	5421006.pn.	1
☐	L32	5537540.pn.	1
☐	L31	5544344.pn.	1
☐	L30	5623673.pn.	1
☐	L29	5671422.pn.	1
☐	L28	5784625.pn.	1
☐	L27	5832299.pn.	1
☐	L26	L10 and L25	4
☐	L25	secure adj3 execution	140
☐	L24	5898843.pn.	1
☐	L23	5956743.pn.	1
☐	L22	5630147.pn.	1
☐	L21	5963738.pn.	1
☐	L20	6009520.pn.	1
☐	L19	6009524.pn.	1
☐	L18	L14 and L17	16
☐	L17	ACPI or (Advanced adj configuration adj2 power adj interface)	326
☐	L16	L14 and L15	0
☐	L15	secure adj execution adj mode$1	1

☐	L14	L12 and L13	160
☐	L13	L10 and L3	306
☐	L12	L7 or L11	7708
☐	L11	((710/260 |710/261 |710/262 |710/263 |710/264 |710/265 |710/266 |710/267 |710/268 |710/269)!.CCLS.)	1364
☐	L10	smi or L9	1292
☐	L9	system adj managment adj interrupt	0
☐	L8	L3 and L7	136
☐	L7	L4 or L5 or L6	6416
☐	L6	((713/200 |713/201 |713/202 |713/300 |713/310 |713/320 |713/321 |713/322 |713/323 |713/324 |713/330 |713/340)!.CCLS.)	4764
☐	L5	((713/100)!.CCLS.)	644
☐	L4	((713/1 |713/2)!.CCLS.)	1527
☐	L3	L2 or L1	418
☐	L2	smm and L1	321
☐	L1	system adj management adj mode$1	418

END OF SEARCH HISTORY